TEXT EQUIVALENCING ENGINE

INVENTORS

Ted Dunning
Bradley Kindig

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority from U.S. Patent Application Serial No. _____ for "Relationship Discovery Engine," filed April 30, 2001 the disclosure of which is incorporated herein by reference. The present application also claims priority from provisional U.S. Patent Application Serial No. 60/201,622, for "Recommendation Engine," filed May 3, 2000, the disclosure of which is incorporated herein by reference.

BACKGROUND

A. Technical Field

[0002] The present invention is related to text equivalencing, and more particularly, to text equivalencing for human language text, genetic sequences text, or computer language text.

B. Background of the Invention

[0003] In many contexts finding equivalent text is desirable. Equivalent texts are two pieces of text that are intended to be exactly the same, at least one of which contains a misspelling or typographical error or an difference in representation such as using a symbolic representation of a word or a different word order. One context where finding equivalent text is desirable is in identifying a song or book by its textual identifiers such as title, artist or author, album name or publisher name, etc. Another such context is in identifying genetic sequences. In these and other contexts, it is desirable to be able to identify equivalent texts because

typographical errors or misspellings can occur. In particular, misspellings occur with a high frequency with foreign-sounding text or names, or when the text is the product of voice transcription. Also, there are times when certain words, such as, "the" are omitted or added erroneously, or a character is replaced by a word or vice versa, for example "&" to "and" or "@" to "at."

[0004] In the context of text equivalencing for textual information related to music, there are many ways the text ends up close, but not exactly the same. For example, when a song is downloaded the information can be typed in manually by the user, thus increasing the chance for error. In the context of a software music player it is generally important to identify the track of music by the textual information. If the textual information cannot be determined it is difficult to identify the music track.

[0005] One method of identifying an equivalent text is by applying a simple set of heuristics to the text and then comparing it to known texts. The simple set of heuristics can overcome problems of variable amounts of white space and omitting or adding words such as "the" at the beginning of an album or book name. However, such heuristics fall short when it comes to typographical errors or misspellings. It is not possible to apply a heuristic for every possible mistyping or misspelling based on every possible pronunciation of every word in the text. The problem is further compounded when the words are not actual language words, but instead are proper names, genetic sequences, acronyms, or computer commands. In those contexts, predicting the mistakes and forming the heuristics is more difficult than when the words are language words.

[0006] What is needed is a system and method of text equivalencing that avoids the above-described limitations and disadvantages. What is further needed is a system and method for text equivalencing that is more accurate and reliable than prior art schemes.

SUMMARY OF THE INVENTION

[0007] The present invention provides a text equivalencing engine capable of determining when one string of characters or text is equivalent or probably equivalent to another string of characters or text. In one embodiment, the string of characters is textual information describing a track of music such as a song title, artist name, album name or any combination of these attributes. In another embodiment, the string of characters is textual information describing a book or magazine such as title of the book or magazine, author name, or publisher. In another embodiment, the text is a genetic sequence or the text is a computer program listing. One skilled in the art will recognize that the techniques of the present invention may be applied to any type of text or character string. The present invention can determine when two strings of characters are equivalent even when there are misspellings or typographical errors that are not addressed by a set of heuristics. The present invention can be used to accurately perform text equivalencing in most cases. Thus, the present invention overcomes the above-described problems of misspellings and typographical errors.

[0008] The text equivalencing is performed by first applying a set of heuristics. The set of heuristics modifies the text to be equivalenced by a set of rules. The rules are designed to eliminate common mistakes such as eliminating or adding whitespace, adding or deleting the

word "the," changing "&" to "and" and vice versa, and a few common misspellings and typographical errors.

[0009] The modified strings are compared to known strings of characters or text. If a match is found, the text equivalencing process is exited successfully. If an exact match is not found, the string of characters is separated into sub-strings. The sub-strings can be of any length. In one embodiment, the sub-strings are typically three characters long and are referred to as 3-grams. In another embodiment, the length of the sub-strings is adaptively determined.

[0010] Any information retrieval technique can be applied to the sub-strings to perform accurate text equivalencing. In one embodiment, the information retrieval technique weights the string of characters and scores other known strings of characters using a technique known as Inverse Document Frequency (IDF) as applied to the sub-strings in each string of characters. If the highest scoring known string of characters scores above a first threshold, t_1, then that known string of characters is accepted as a match to the string of characters and all other known strings are considered not to be matches. If one or more of the scores is between the first threshold, t_1, and a second threshold, t_2, the string of characters is accepted as a match after a user manually confirms the match and all known strings scoring below t_2 are discarded. If no known strings score above t_2 and there is a set of known strings scoring between the second threshold, t_2, and a third threshold, t_3, all strings scoring in this range are presented to the user as options and all strings scoring below t_3 are discarded. The user can select one of the strings as a match. If the scores are all below the third threshold, t_3, the strings are ignored and a match is not determined. Once a match is determined, the matching string can be updated to reflect the fact that the string of characters is equivalent to it.

[0011] As can be seen from the above description, the present invention may be applied to many different domains, and is not limited to any one application. Many techniques of the present invention may be applied to text equivalencing in any domain.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Fig. 1 is a block diagram of a functional architecture for text equivalencing.

[0013] Fig. 2 is a flow diagram of a method of text equivalencing in accordance with the present invention.

[0014] Fig. 3 is a flow diagram of a method of database update in accordance with the present invention.

[0015] Fig. 4 is a flow diagram of a method of grouping characters to form sub-strings.

[0016] Fig. 5 is a flow diagram of a method of scoring text and strings of characters using the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0017] The following description of preferred embodiments of the present invention is presented in the context of a text equivalencing engine for any string of characters. In some embodiments, it may be implemented for use in identifying mistyped or misspelled song titles, artist names or album titles for an Internet-based jukebox or for identifying genetic sequences. One skilled in the art will recognize that the present invention may be implemented in many other domains and environments, both within the context of text equivalency, and in other contexts. Accordingly, the following description, while intended to be illustrative of a particular implementation, is not intended to limit the scope of the present invention or its applicability to other domains and environments. Rather, the scope of the present invention is limited and defined solely by the claims.

[0018] Now referring to Fig. 1, there is shown a block diagram of a functional architecture for text equivalencing. Also, referring to Fig. 2, there is shown a method of text equivalencing. Text equivalencing is useful for determining when one string of characters is equivalent to another string of characters. A character may be any letter, number, space, or other symbol that can be inserted into text. In an Internet jukebox or personalized radio system, it is typically important to be able to identify a track of music by its title, artist, and album name because such a system makes recommendations based on a history of listener behavior. In other domains it is also important to be able to determine when one piece of text is equivalent to another. For example, in genetic sequencing it is useful to determine when two genetic

sequences are equivalent. Also, in computer programming, text equivalencing can be an important tool as a reverse engineering tool for documenting large programs.

[0019] The text equivalencing engine 140 takes a string of characters 105 and determines a text equivalent 135 to the string of characters. A heuristics module 110 modifies the character string by applying a set of rules to the string of characters 205. In one embodiment, there are typically about 50 heuristics that are applied. However, there could be fewer or greater than 50 heuristics applied depending on the system and the type of text.

[0020] In one embodiment, the heuristics include omitting whitespace, adding or deleting the word "the," converting "&" to "and" and vice versa, and any other heuristic as may be appropriate or desired. Also, if the string of characters is words in a human language, a heuristic could be applied that changes the order of the words in the string such as in the modification of "Einstein, Albert" to "Albert Einstein." In one embodiment, once the heuristics have been applied, a database containing known character strings is searched by comparator module 117 to find any exact match to the character string or any of the new characters strings created by modifying the character string in accordance with the heuristics 205. If such a match 119 is found 215, text equivalencing engine 140 ceases searching for equivalents and the database update module 145 performs database update 235, a process by which the set of known character strings are updated to indicate that the string of characters is an equivalent text.

[0021] If no match 118 is found 215, the character strings 115 are divided into sub-strings. Sub-strings 125 are formed in the sub-string formation module 120 by selecting frequently occurring groups of characters 225. The formation of sub-strings is further described below in reference to Fig. 4. In one embodiment, the sub-strings 125 are 3-grams. A 3-gram is a

sub-string 125 that is three characters in length. In another embodiment, the sub-strings 125 can be of any length and are selected by frequency in the corpus of known strings.

[0022] The sub-strings 125 are input into an information retrieval module 130. The information retrieval module can perform any accurate information retrieval technique to retrieve an equivalent text 135 to the character string 105 from a database of known character strings 230. Any information retrieval technique may be implemented. Examples of such information retrieval techniques are discussed in G. Salton & M. McGill, "Introduction to Modern Information Retrieval," McGraw-Hill, 1983 and P. Willet and K. Sparck Jones (ed), "Readings in Information Retrieval," Morgan Kaufmann, 1997.

[0023] In one embodiment, module 130 employs an information retrieval technique 230 using a score relative to each known string of characters in the database. The score is based on a term-weighting scheme described below in reference to Fig. 5. After the information retrieval module 130 determines an equivalent text or a set of candidate equivalent texts 230, database update module 145 typically updates a database of known equivalent texts to indicate that the string of characters is equivalent 235. The database update module 145 is further described below with the reference to Fig. 3.

[0024] The most likely equivalent string of characters is determined by the retrieval score as produced by the information retrieval module 130. The score is evaluated based on three threshold values, t_1, t_2, and t_3. Typically, $t_1 > t_2 > t_3$, creating four regions, greater than t_1, between t_1 and t_2, between t_2 and t_3, and less than t_3. If the highest of the of the scores is greater than a first predetermined threshold value, t_1, the text with that score is accepted as an equivalent to the string of characters without any manual intervention. If the highest of the scores is

between the first threshold, t_1, and a second predetermined threshold, t_2, that text is accepted as an equivalent to the string of characters after a user manually confirms the match. If there are one or more scores between the second threshold value, t_2, and a third predetermined threshold value, t_3, each text scoring in that range is presented to the user. One piece of text is accepted as an equivalent to the string of characters only after being selected by the user. If all the scores are below the third threshold, t_3, no equivalent is found. In one embodiment, the threshold values are empirically determined to optimize the amount of manual labor needed to build a table of text equivalences.

[0025] Now referring to Fig. 3, there is shown a method of performing database update 235. Database update is performed 235 when an equivalent but not identical text has been found and accepted 305. If an equivalent text is determined, the database update module 145 performs database update 235. In one embodiment, database update involves updating 310 the database that stores the known strings of characters to indicate that the string of characters is an equivalent text. The database update module 140 takes into consideration any user confirmation or selection that is made in accepting the document as equivalent text. Thus, if the same string of characters 105 is input into the text equivalencing engine 140 on a subsequent run, there will be no need to run through the entire text equivalencing engine 140.

[0026] Now referring to Fig. 4, there is shown a method of forming variable length sub-strings from a string of characters. Sub-strings are formed from a series of characters in a given string of characters by extending the sub-strings based on the frequency of occurrence of the extended sub-strings. In one embodiment, there are two thresholds that together define whether a sub-string is considered "frequently appearing" in each document. The threshold

values are chosen such that the words yield an accurate, fast, and memory-efficient result of identifying a character string. The first threshold, for the purpose of sub-string formation, is a minimum number of appearances of a sub-string in a document. In one embodiment, the second threshold, for the purpose of sub-string formation, is a maximum number of appearances of a sub-string in a document. A sub-string is considered to be "frequently appearing" if its frequency lies between the thresholds for sub-string formation. In an alternate embodiment, only one of the two thresholds for sub-string formation is used.

[0027] When sub-strings are formed, initially, each character is assumed to be a sub-string 405. Then, in one embodiment of the present invention, the system looks for extensions of frequently appearing sub-strings 410 formed by adding one character. These are also considered sub-strings. In one embodiment, the system then looks for frequently appearing 3-grams 415 (a three letter word). In one embodiment, the system then looks for frequently appearing 4-grams 415. Finally, the system looks for frequently appearing n-grams 415, where n is any positive integer value greater than 2. In one embodiment, in the event of overlap between a substring and any longer sub-string, the system favors the longer sub-string 425 and presumes that the shorter did not occur. The result is a series of the longest frequently appearing n-grams. In another embodiment, the system favors 3-grams, instead of favoring the longer sub-strings. In another embodiment, frequency of occurrence is determined relative to a separate large corpus of strings.

[0028] The following is a hypothetical example of sub-string formation for an arbitrary string of letters from the alphabet. The letters could be any kind of symbol, but for this hypothetical are only alphabetical letters. Normally the entire corpus of known strings would be

used to build a dictionary of possible sub-strings, but this example is a short string in order to simplify the description.

a g e c q r t l m s d s p l c q r r a g e t l p r a g e s l

[0029] Assume for this hypothetical example that the threshold for purposes of sub-string formation is 2. Each character is initially taken to be a sub-string 405. The character pair "ag" appears 3 times, making it a frequently appearing sub-string 410, according to the threshold value for sub-string formation. The 3-gram, "age," appears 3 times making it a frequently appearing sub-string 415. The 4-gram "rage" also occurs twice. The sub-strings "agec," "aget," and "ages" each only appear once. Therefore no more characters should be added and the 4-gram "rage" is the longest n-gram that can be formed to meet the frequently appearing sub-string criteria. The same process could be performed on the second character in the string, "g." However, since "ge" overlaps with "rage", the "ge" sub-string would be eliminated when favoring longer sub-strings 420.

[0030] The same process could be performed to form the sub-string "cqr" as to form the sub-string "rage." After the system finished forming sub-string on this string the following sub-strings would be formed: "rage" appearing 2 times, "cqr" appearing 2 times, "tl" appearing 2 times, and a number of 1-grams appearing various numbers of times. Thus the decomposition of the larger string into sub-strings would be:

a / g / e / c q r / t l / m / s / d / s / p / l / c q r / r a g e / t l / p / r a g e / s / l

[0031] An information retrieval technique is performed on the sub-strings to find an equivalent text. In one embodiment, the information retrieval technique involves the sub-strings and scoring known pieces of text that are also divided into sub-strings. In one embodiment, the

present invention employs an inverse document frequency method for scoring sub-strings in the string of characters and known text.

[0032] Referring now to Fig. 5, there is shown a flow diagram of a method of weighting sub-strings and scoring texts according to the present invention. The method illustrated in Fig. 5 is shown in terms of matching sub-strings in a text equivalencing system. One skilled in the art will recognize that the method may be adapted and applied to many domains and techniques.

[0033] A total number of texts N is determined 504. The system also determines a text frequency for each sub-string j (the number of times sub-string j occurred, or $TF_j = \sum(k_{ij} > 0)$ where k_{ij} is the number of times that text i contains sub-string j) 504.

[0034] Weights are computed for query sub-strings and for sub-strings of each equivalent text. The weights are computed according to a product of up to three components: l = the number of times a sub-string appears in the string of characters; g = the number of times a sub-string appears in the known text; and n = a normalizing factor based on the entire weight vector.

[0035] The first weighting factor, l, is a local weighting factor. It represents the frequency of the sub-string within a string of characters. It may be represented and defined according to the following alternatives as well as others known in the art of information retrieval:

$l_T = k_{ij}$ = Number of times sub-string j occurs in the string i; or

$l_L = \log(k_{ij} + 1)$; or

$l_X = 1$ (a constant, used if this weighting factor is not to be considered).

l may be adjusted to account to optimize performance for different kinds of strings.

[0036] The second weighting factor, g, represents the frequency of the sub-strings within all the known texts. It may be represented and defined according to the following alternatives as well as others known in the art of information retrieval:

$g_I = \log \frac{N+1}{TF_j + 1}$ (inverse text frequency, i.e., the log of the total number of texts divided by the text frequency for sub-string j); or

$g_X = 1$ (a constant, used if this weighting factor is not to be considered).

g may be adjusted in a similar manner as is l to optimize performance.

[0037] The third weighting factor, n, represents a normalizing factor, which serves to reduce the bias that tends to give long texts higher scores than short ones. Using a normalizing factor, a short, specifically relevant text should score at least as well as a longer text with more general relevance. n may be represented and defined according to the following alternatives as well as others known in the art of information retrieval:

$$n_C = \frac{1}{\sqrt{\sum_j (l_j)^2 (g_{ij})^2}}; \text{ or}$$

$n_X = 1$ (a constant, used if this weighting factor is not to be considered).

[0038] By employing the above-described combination of three weighting factors in generating weights for sub-strings and scores for texts, the present invention avoids the problems

of overstating frequently appearing sub-strings and overstating coincidental co-occurrence. If a sub-string is frequently occurring, the second weighting factor will tend to diminish its overpowering effect. In addition, the effect of coincidental co-occurrence is lessened by the normalization factor.

[0039] In one embodiment, the system of the present invention generates scores as follows. Each sub-string j 502 in query string 501 i is weighted 503 using the general formula $q_{ij} = l_{ij}g_jn_i$, where l, g, n are defined above. The weight w_{ij} for each sub-string 505 j in each known text i is obtained 506 using the general formula $w_{ij} = l_{ij}g_jn_i$, where l, g, n are defined above. The specific options for defining l, g and n can be chosen based on the desired results. In one embodiment of the present invention, it is desired to preserve diversity, to give rarity a bonus, and to normalize. In that embodiment, the weighting options L, I, and C may be chosen. Weighting option L acts to preserve diversity, option I acts to give variety a bonus, and C acts to normalize the results. Thus, in that embodiment the weighting would be equal to $l_Lg_In_C$. In other embodiments, different weighting options may be used. In one embodiment, the present invention uses the same weighting for the query sub-strings as it does for scoring the known texts. In an alternative embodiment, different weighting options can be chosen for the query sub-strings and for scoring the known texts.

[0040] A score for a text is determined by taking the dot product 507 of the query vector and the document vector $\sum_j q_{kj}w_{ij}$. In one embodiment of the present invention, the above-described weighting factors are used to create the vector terms in order to improve the results of the scoring process. The score falls into one of four categories 508 and the results are

output 509, depending on the value of the score, as described below. If the highest score is greater than a first threshold value, t_1, the text is accepted as an equivalent text. In one embodiment, this range of scores is considered the gold range. If the highest score is between the first threshold value, t_1, and a second threshold value, t_2, that text is accepted as an equivalent text after manual user confirmation. In one embodiment, this range of scores is considered the silver range. If one or more scores falls in a range between the second threshold value, t_2, and a third threshold value, t_3, each of the texts scoring in that range are presenting to the user. The user can select a text to be accepted as equivalent text. In one embodiment, this range of scores is considered the bronze range. If all the texts score below the third threshold value, t_3, all of the texts are ignored and an equivalent text is not determined.

[0041] In one embodiment, the threshold values are adjusted empirically to give the desired accuracy. In one embodiment, the desired accuracy is about 100 % in the gold range. In one embodiment, the desired accuracy in the silver range is 70 – 80 % the top one text equivalent and 90 % in the top five text equivalents. In one embodiment, the bronze range is about 20 – 40% accurate and 50% in the top five.

[0042] Once the query sub-strings have been weighted and the known texts have been scored by computing the dot product of two vectors, the text or texts with the highest score or highest scores are determined. An equivalent text is selected using the threshold values described above.

[0043] From the above description, it will be apparent that the invention disclosed herein provides a novel and advantageous system and method for text equivalencing. The foregoing discussion discloses and describes merely exemplary methods and embodiments of the

present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. For example, the invention may be applied to other domains and environments, and may be employed in connection with additional applications where text equivalencing is desirable. Accordingly, the disclosure of the present invention is intended to illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.